Cryptographic Protocol Analysis

Jonathan Millen
SRI International

Cryptographic Protocols in Use

Cryptographic protocol: an exchange of messages over an insecure communication medium, using cryptographic transformations to ensure authentication and secrecy of data and keying material.

Applications: military communications, business communications, electronic commerce, privacy

Examples:

- Kerberos: MIT protocol for unitary login to network services
- SSL (Secure Socket Layer, used in Web browsers), TLS
- IPSec: standard suite of Internet protocols due to the IETF
  - ISAKMP, IKE, JFK, ...
- Cybercash (electronic commerce)
- EKE, SRP (password -based authentication)
- PGP (Pretty Good Privacy)

The Security Threat:
Active Attacker (Dolev - Yao model)

QuickTime™ a
TIFF (Uncompress
are needed to se

Attacker **can**:
- -intercept all messages
- -modify addresses and data

Attacker **cannot**:
- -encrypt or decrypt without the key (ideal encryption)

A Simple Example

The Needham-Schroeder public-key handshake

(R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," CACM, Dec., 1978)

A → B: {A, Na}pk(B)

B → A: {Na, Nb}pk(A)

A → B: {Nb}pk(B)

This is an "Alice-and-Bob" protocol specification

Na and Nb are nonces (used once)

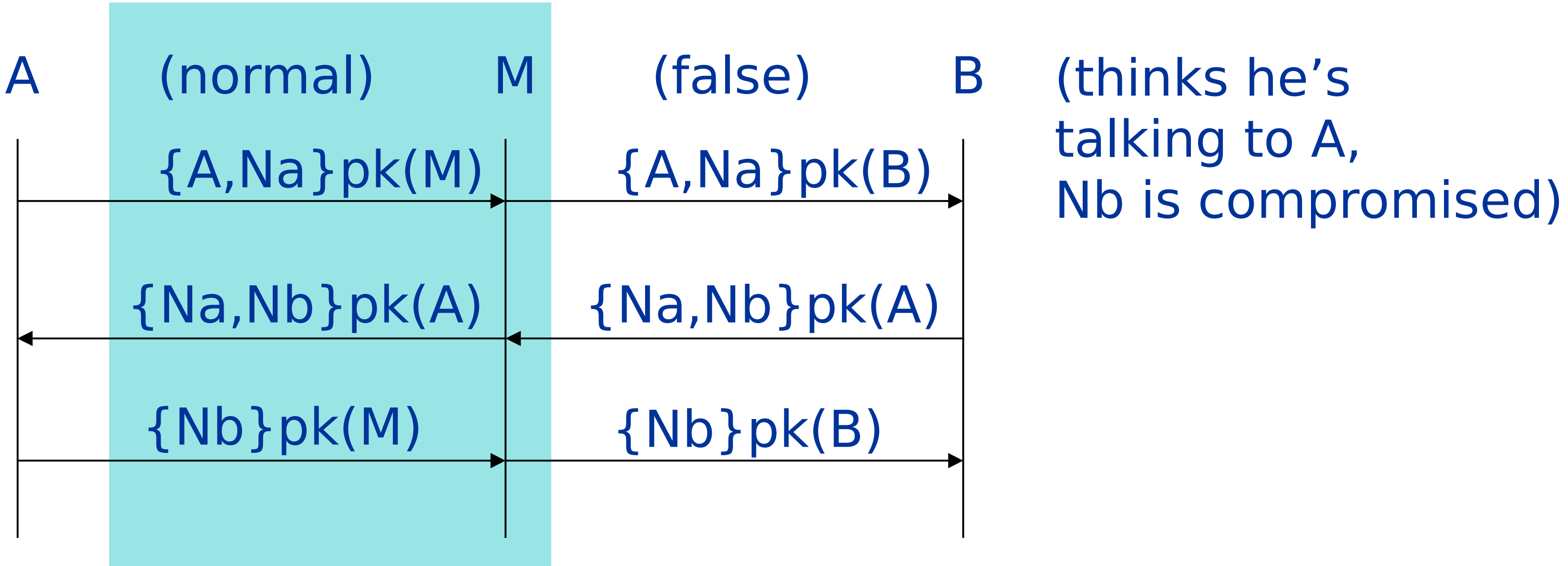
pk(A) is the public key of A

A and B authenticate each other, Na and Nb are secret

The protocol is vulnerable...

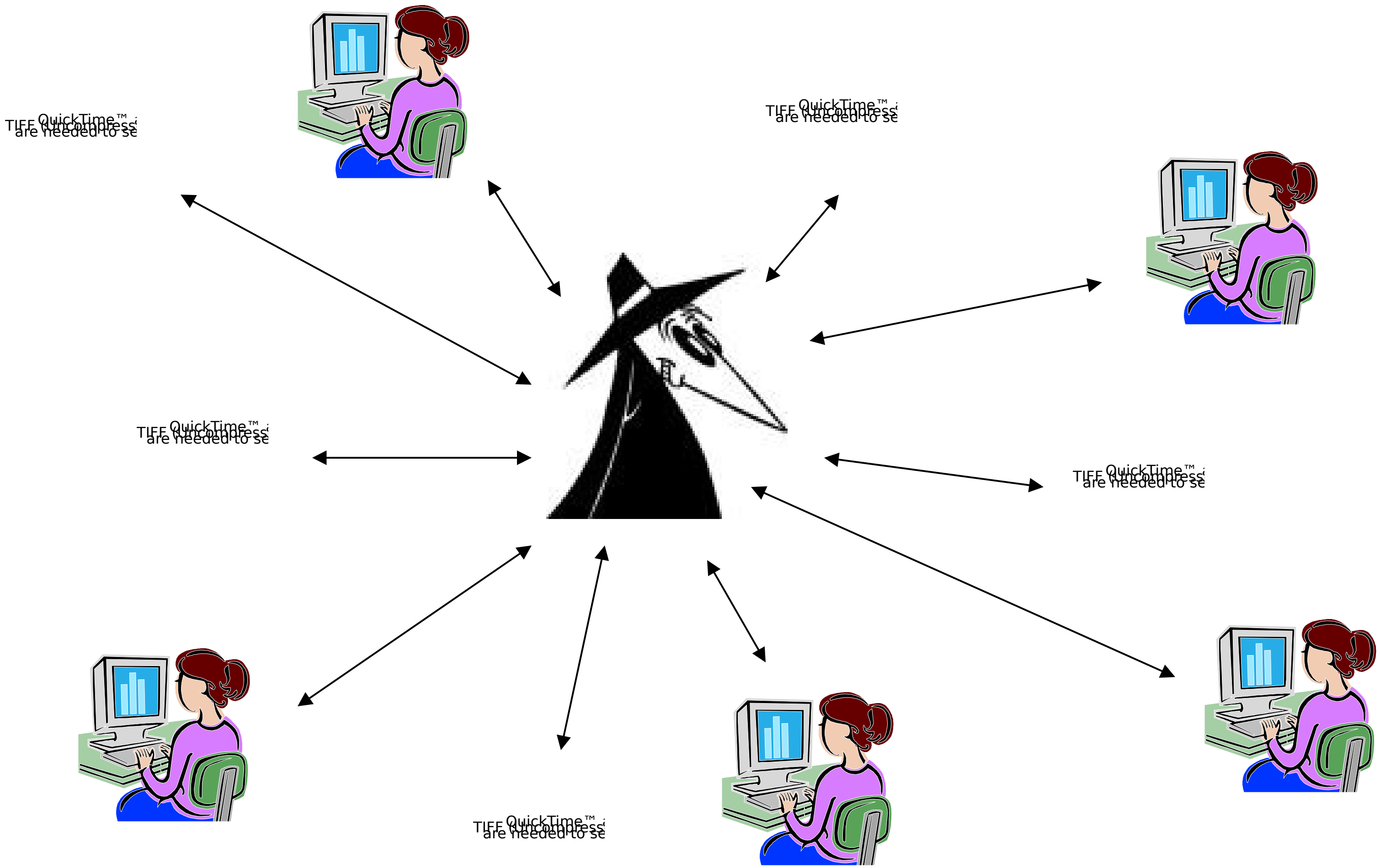
The Attack

A malicious party M can forge addresses, deviate from protocol



Lowe, "Breaking and Fixing the Needham-Schroeder Public Key Protocol Using FDR," Proc. TACAS 1996, LNCS 1055

Why Protocol Analysis is Hard



What Makes Protocol Analysis Hard?

The attacker.

Unbounded number of concurrent sessions.

Recursive data types.

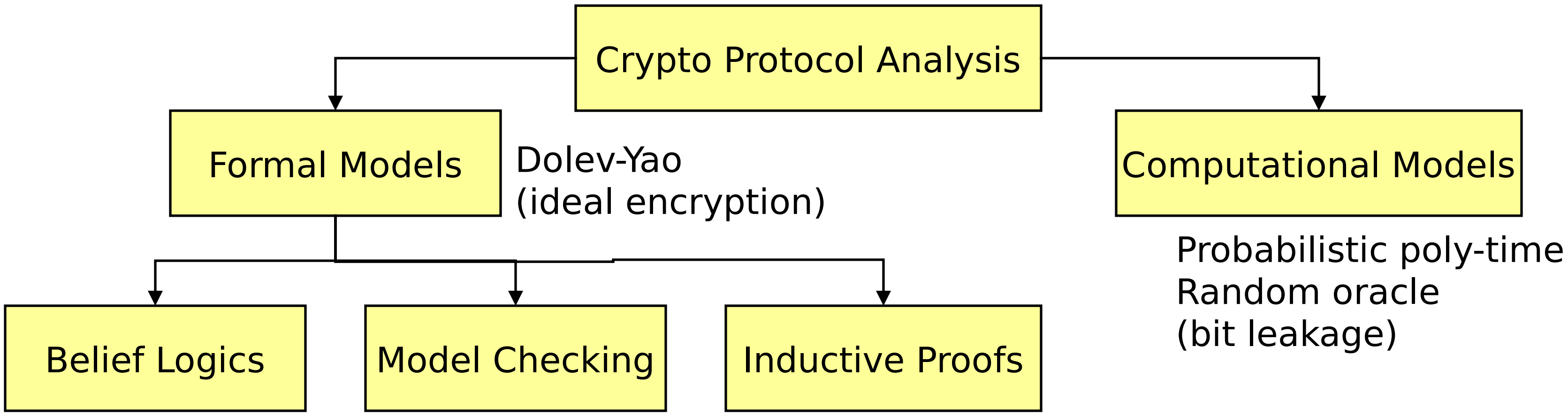
$\{a,b,c, \ldots \} = \{a, \{b, \{c, \ldots \}\ldots\}$

$\{\ldots\{\{\{a\}k_1\}k_2\}k_3 \ldots$

Infinite data types (nonces)

$n_1, n_2, n_3, \ldots$

Crypto Protocol Analysis



Belief Logics

Origin: Burrows, Abadi, and Needham (BAN) Logic (1990)
Modal logic of belief (“belief” as local knowledge)
Special constructs and inference rules
  e.g., **P sees X** (P has received X in a message)
Protocol messages are “**idealized**” into logical statements
Objective is to prove that both parties share common beliefs
Example inference rule:

$$\frac{\textbf{P believes fresh(X), P believes Q said X}}{\textbf{P believes Q believes X}}$$

Implicit assumption that secrets are protected!
Good for authentication proofs, but not confidentiality

Model Checking Tools

State-space search for reachability of insecure states

- History: back to 1984, Interrogator program in Prolog
- Meadows' NRL Protocol Analyzer (NPA), also Prolog
- Early Prolog programs were interactive
- Song's Athena is recent, automatic

General-purpose model-checkers applied

- Searched automatically given initial conditions, bounds
- Roscoe and Lowe used FDR (model-checker for CSP)
- Mitchell, et al used Murphi
- Clarke, et al used SMV
- Denker, et al used Maude

Can only search a finite state space

Inductive Proofs

Approach: like proofs of program correctness

- Induction to prove secrecy invariant

General-purpose specification/verification system support

- Kemmerer, using Ina Jo and ITP (1989) (the first)
- Paulson, using Isabelle (1997) (the new wave)
- Dutertre and Schneider, using PVS (1997)
- Bolignano, using Coq (1997)

Can also be done manually

- Schneider, in CSP; Guttman, et al, in strand spaces
- Contributed to better understanding of invariants
- Much more complex than belief logic proofs

Full guarantee of correctness (with respect to model)

- Proofs include confidentiality
- No finiteness limits

Undecidable in General

Reduction of Post correspondence problem

- Word pairs u_i, v_i for $i = 1, \ldots, n$
- Does there exist $u_{i1} \ldots u_{ik} = v_{i1} \ldots v_{ik}$?
- No general algorithm to decide

Protocol:

- Compromises secret if solution exists
- Attacker can feed output of one instance to input of another
- Attacker cannot read or forge messages because of encryption
- Messages are unbounded

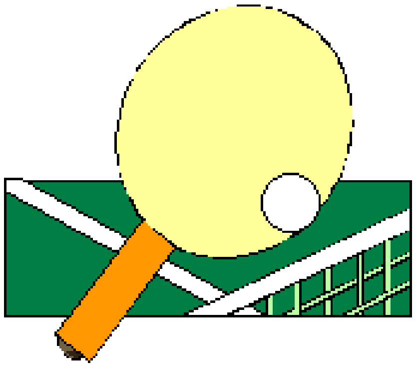
Initial party

send $\{\varepsilon, \varepsilon\}K$

The i^{th} party

receive $\{X,Y\}K$
if $X = Y \neq \varepsilon$, send **secret**
else send $\{Xu_i, Yv_i\}K$

A Decidable-Security Version: Ping Pong Protocols (Dolev-Yao '83)



- Abstract public-key encryption E_a, decryption D_a per party
- Reduction $D_a E_a M = M$
- Protocol accumulates operators on an initial message M
- Attacker intercepts messages and applies operators also

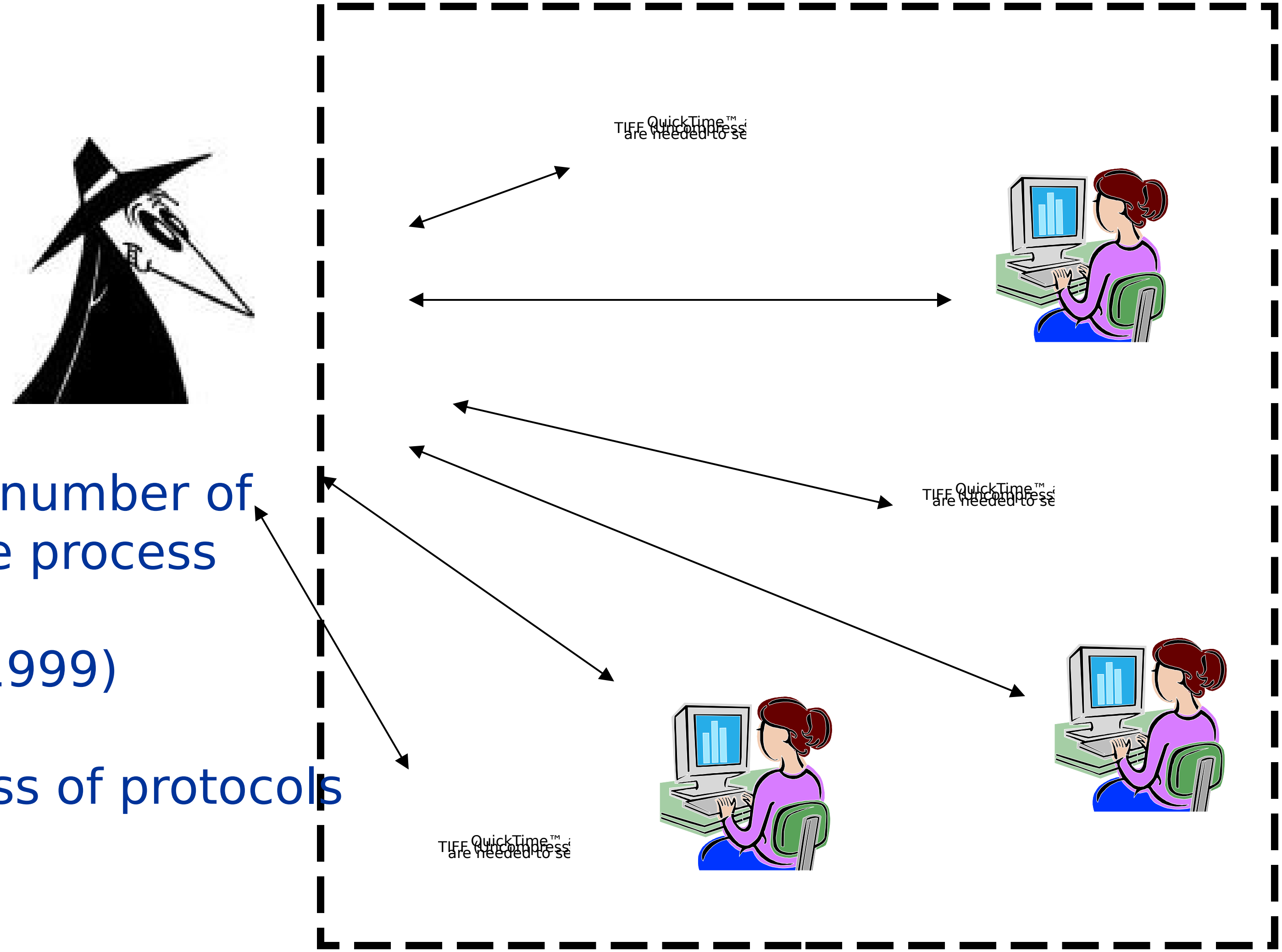
$$M \xrightarrow{E_a M} E_a M$$

$$E_a E_a M \xleftarrow{E_a E_a M} E_a$$

$$E_b D_a \xrightarrow{E_b D_a E_a E_a M} E_b E_a M$$

Secrecy of M decidable in linear time

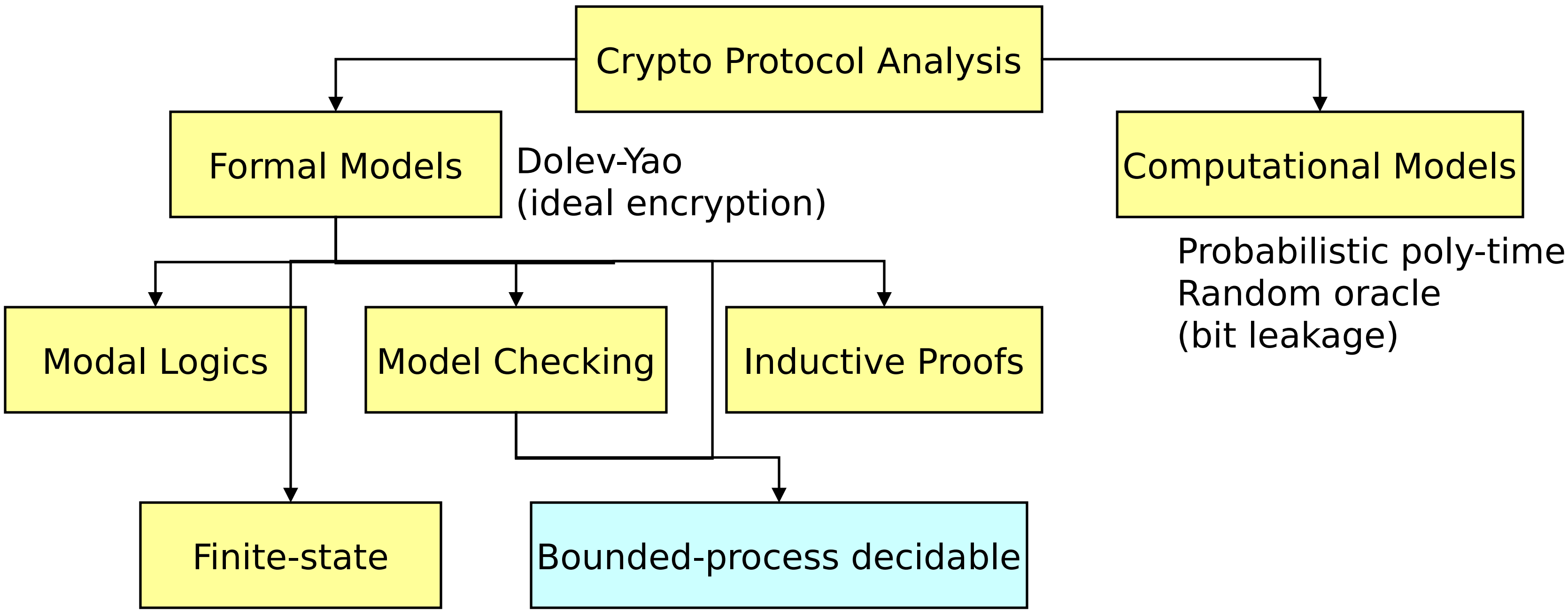
Bounded-Process Decidability



Limit the number of legitimate process instances
(Huima, 1999)

Large class of protocols

Crypto Protocol Analysis



Antti Huima, 1999
Symbolic states
(extended
abstract)

Inspired subsequent work:
Amadio-Lugiez
Boreale
Fiore-Abadi
Rusinowitch-Turuani (NP-complete)
Millen-Shmatikov (constraint solver)

Constraint Solving

Parametric strand specification

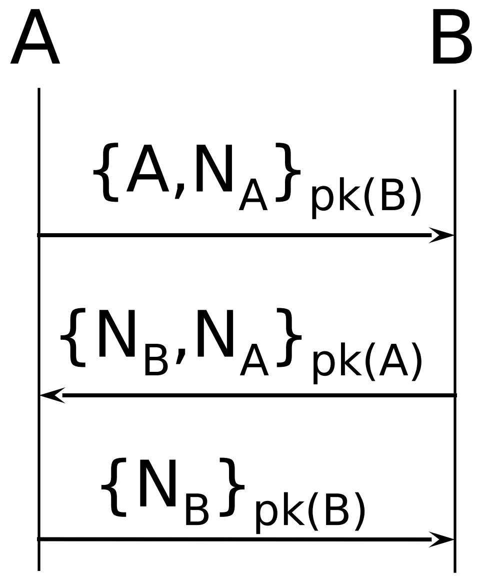
Finite scenario setup

Generating constraint sets

Solve constraint set (finds attack) or prove unsolvable (secure)

Parametric Strand Specification

Protocol



A-role strand

+
$\{A,N_A\}_{pk(B)}$

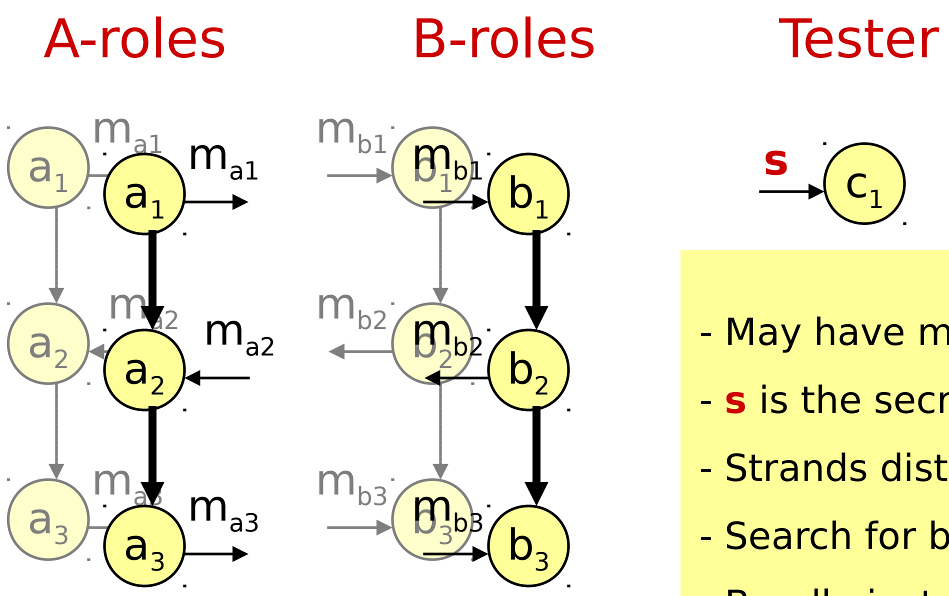
-
$\{N_B,N_A\}_{pk(A)}$

+
$\{N_B\}_{pk(B)}$

B-role strand

- Strand: node sequence
- Node is directed message term
- Role has variables in terms

Semibundle Scenario

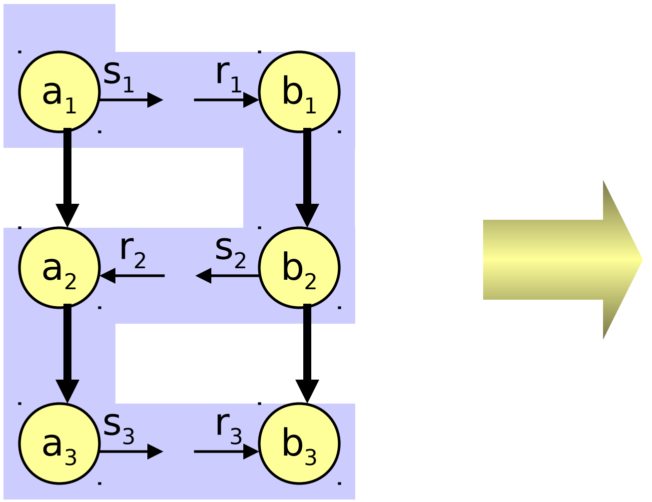


- May have multiple role instances
- **s** is the secret (skolem constant)
- Strands distinguished by constant nonces
- Search for bundle
- Bundle instantiates variables

Bundle: every received message is computable by an attacker (original "bundle" includes explicit attacker operation strands)

Constraint Set Generation

Enumerate all linear node orderings consistent with strands



Constraints

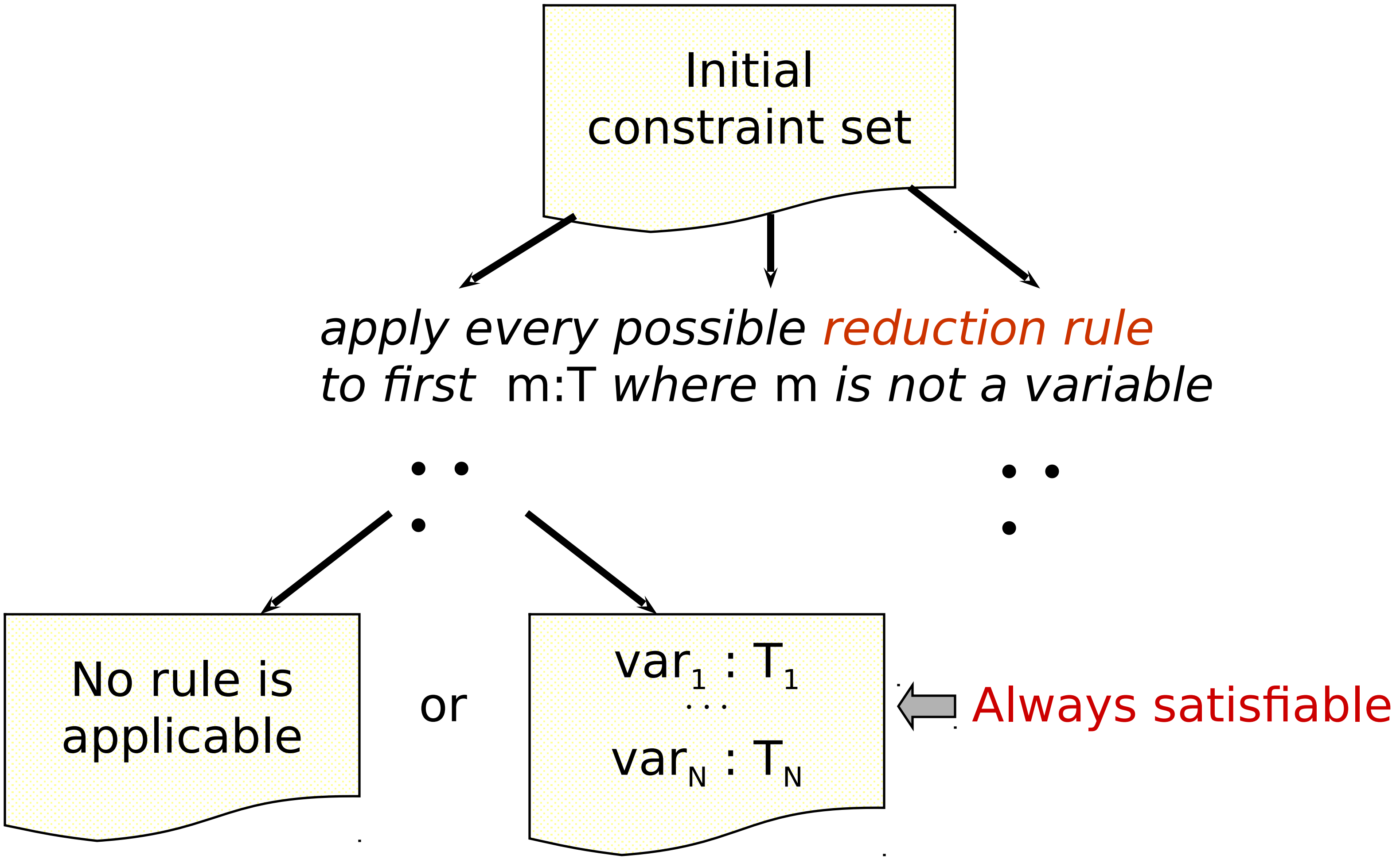
r_1: T_0 , s_1
r_2: T_0 , s_1 , s_2
r_3: T_0 , s_1 , s_2, s_3

m: T *means* m can be computed from T

received messages have computability constraint

(T_0 is set of terms known initially to attacker)

Reduction Tree



A constraint set is solvable if it is reducible to a satisfiable set

Analysis Rule Example

$$\frac{m : \{t_1, t_2\}, T}{m : t_1, t_2, T} \ (\textit{split})$$

Synthesis Rule Example

$$\frac{\{m\}_k : T}{\begin{array}{c} m : T \\ k : T \end{array}} \; (enc)$$

Unification Eliminates a Constraint

$$\frac{m : t, T}{-}\ (un)$$

Unify left side term with some term on right

Instantiate variables if necessary - part of solution

Encryption Decomposition

$$\frac{m : \{t\}_k \,, T}{\begin{array}{l} k : \{t\}_k{}^*, T \\ m : t, k, T \end{array}} \; \mathit{(sdec)}$$

***Encrypted term is marked to avoid looping**

Implementation

Prolog Program

Standard Edinburgh Prolog

(can use public domain SWI or XSB)

Short - three pages

Fast - 50,000 interleavings/minute normally

Easy protocol specification

NSPK for Prolog Solver

```
strand(roleA,A,B,Na,Nb,[
  recv([A,B]),
  send([A,Na]*pk(B)),
  recv([Na,Nb]*pk(A)),
  send(Nb*pk(B))
]).

strand(roleB,A,B,Na,Nb,[
  recv([A,Na]*pk(B)),
  send([Na,Nb]*pk(A)),
  recv(Nb*pk(B))
]).

strand(test,S,[recv(S)]).
```

$A \rightarrow B$: {A, Na}pk(B)

$B \rightarrow A$: {Na, Nb}pk(A)

$A \rightarrow B$: {Nb}pk(B)

- Capital letters are variables
- Originated variables must be nonce
- Principals are not originated

(Originated: appearing first in a send)

Scenario Semibundle and Query

```
nspk0([Sa,Sb,St]) :-
  strand(roleA,_A,_B,na,_Nb,Sa),
  strand(roleB,a,b,_Na,nb,Sb),
  strand(test,nb,St).

:- nspk0(B),search(B,[]).
```

- The secret and the principals sharing it are instantiated
- Other nonces are instantiated in originator strand

- Authentication test is also possible

Search Result

?- nspk0(B),search(B,[ ]).
Starting csolve...
 Try 1 Try 2 Try 3 Try 4
Simple constraints:

Trace:
recv([a, e])
send([a, na]*pk(e))
recv([a, na]*pk(b))
send([na, nb]*pk(a))
recv([na, nb]*pk(a))
send(nb*pk(e))
recv(nb*pk(b))
recv(nb)

e *is the attacker*

a → e: {a, na}pk(e)
? → b: {a, na}pk(b)
b → a: {na, nb}pk(a)
? → a: {na, nb}pk(a)
a → e: {nb}pk(e)
? → b: {nb}pk(b)
? → ?: nb

Other Resources

Information on CAPSL web site:

http://www.csl.sri.com/~millen/capsl

ACM CCS-8 paper, "Bounded-process cryptographic protocol analysis"

Prolog constraint solver program and NSL example

Bibliography